



jonathan.decker@uni-goettingen.de

Jonathan Decker

Certificates and PKI

Simple in Principle - Complicated in Praxis

Table of contents

- 1 What are Certificates and PKI
- 2 Terminology
- 3 Certificates in the Wild
- 4 Certificate File Formats
- 5 Public Key Infrastructure

Overview

- We use complex math to build a cryptographic system
- Enables TLS (and HTTPS)
- Universally applicable in networks
- Identify devices and code
- Build on top of public-private-key cryptography
- Simple concept, complicated standards

<https://www.math.auckland.ac.nz/~sgal018/crypto-book/crypto-book.html>

Public-Private-Key Cryptography

- Also called *Asymmetric Cryptography*
- Use math to generate key pairs
- Encrypt with one
 - ▶ Decrypt with the other
- Encrypted messages give no hint on the key itself
- Public key can be shared
- Private key must stay private
- Use someone's public on a message
 - ▶ It can only be read with the private key

Notes on Asymmetric Cryptography

- It is slow compared to symmetric crypto
 - ▶ It makes sense if you see the math
- Commonly used to establish a shared secret between client and server
 - ▶ Shared secret enables symmetric cryptography
- Verifying the receiver across a network is taken for granted nowadays
- Thanks number theory

Who makes the Terminology - The IETF

- IETF (Internet Engineering Task Force)
- Founded in 1986
- Develop **voluntary** standards
- Publishes technical documents RFCs
- RFC (Request For Comments)
 - ▶ Historical acronym now RFC means RFC

<https://www.ietf.org/about/introduction/>

RFC 4949 - 1/3

■ Entity

- ▶ *“An active part of a system – a person, a set of persons (e.g., some kind of organization), an automated process, or a set of processes – that has a specific set of capabilities.”*
- ▶ Anything that exists logically or conceptually
 - Your computer, your code, you yourself
- ▶ Can be a server, software package, email contact

■ Identity

- ▶ *“The collective aspect of a set of attribute values (i.e., a set of characteristics) by which a system user or other system entity is recognizable or known.”*
- ▶ Do not confuse with *Identifier*
- ▶ Identifier is a unique key representing an identity

RFC 4949 - 2/3

■ Claim

- ▶ An entity may declare an attribute
- ▶ “My name is Max Mustermann”
- ▶ Another entity may **Authenticate** a claim

■ Authentication

- ▶ Process of confirming the truth of a claim
- ▶ When you login, you claim to be a user
 - And verify the claim by giving your password

■ End Entity or End User

- ▶ *“A system entity that is the subject of a public-key certificate and that is using, or is permitted and able to use, the matching private key only for purposes other than signing a digital certificate; i.e., an entity that is not a CA.”*

RFC 4949 - 3/3

■ Certificate Authority (CA)

- ▶ An entity that issues certificates to subscribers

■ Subscriber

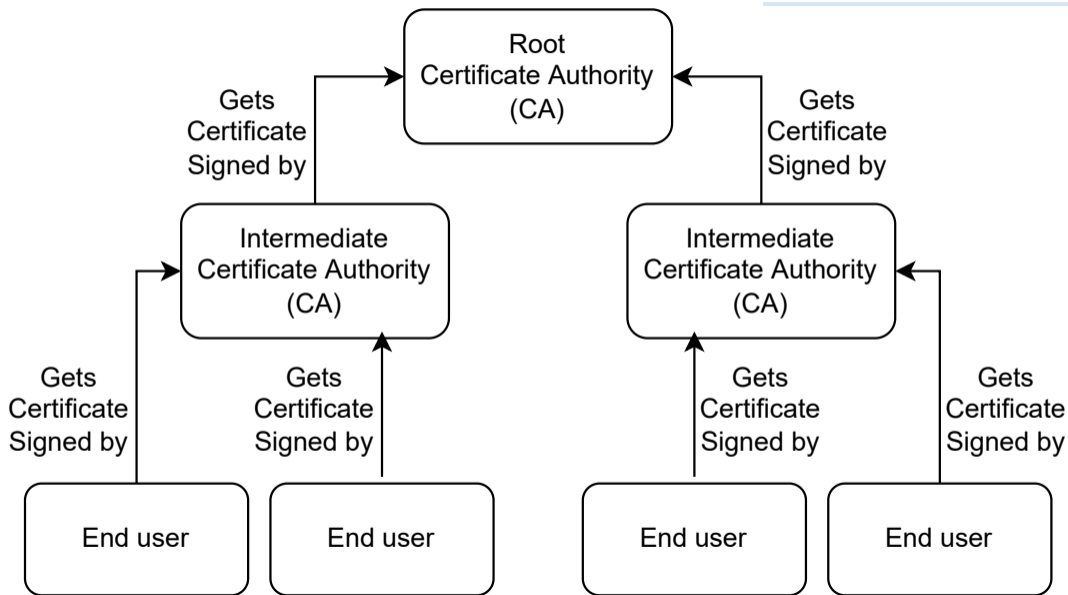
- ▶ *"A user that is registered in a PKI and, therefore, can be named in the "subject" field of a certificate issued by a CA in that PKI."*

■ Public-Key Certificate

- ▶ *"A digital certificate that binds a system entity's identifier to a public key value, and possibly to additional, secondary data items."*
- ▶ A public and metadata (hostname, expiry date, etc.)

■ Root Certificate

- ▶ *"A certificate for which the subject is a root."*
- ▶ *"The self-signed public-key certificate at the top of a certification hierarchy."*



The gwdg.de Certificate

shell

```
echo | openssl s_client -connect gwdg.de:443 2>/dev/null | \
openssl x509 --text
```

The gwdg.de Certificate

Find from top to bottom:

- Unique Serial Number
- Issuer metadata
- Validity range
- Subject metadata
 - ▶ Including encoded public key
- X509v3 extensions
 - ▶ Extended Key Usage
 - ▶ Basic Constraints
 - ▶ CRL Distribution Points
 - ▶ Authority Information Access
 - ▶ CT Precertificate SCTs
 - ▶ Subject Alternative Name
- Signature Algorithm

```
openssl x509 -cert gwdg.de-443-PublicKey1.1\n-----\nCertificate:\n  Version: 3 (0x2)\n  Serial Number:\n    6516e210-9d0d-a216784576c45d-984f61621693\n  Signature Algorithm: sha256WithRSAEncryption\n  Issuer: C = DE, ST = Greater Manchester, L = Salford, E = Sectigo Limited, OU = Sectigo RSA Organisation Validation Services\n  Server CA\n  Validity:\n    Not Before: Feb  2 08:00:00 2023 GMT\n    Not After: Feb  2 23:59:59 2024 GMT\n  Subject: C = DE, ST = North-Rhein, P = Gesellschaft für Cyber wissenschaftliche Datenverarbeitung ghd, OU = gwdg.de\n  Subject Public Key Info:\n    Public Key Algorithm: rsaEncryption\n    rsa Public-Key: (2048 bit)\n    Modulus:\n      00:1f:50:5d:3d:00:08:05:08:5b:19:72:00:78:01:04:05:08:\n      40:27:4d:52:72:78:7d:18:74:5a:31:21:05:21:04:0f:4c:\n      4d:9a:29:72:44:18:1a:7a:59:30:45:01:41:18:42:4e:\n      08:18:50:47:9a:02:02:72:02:00:04:02:01:0a:20:04:\n      2e:1c:1b:00:7b:53:03:02:42:51:7c:08:04:0c:\n      00:1a:00:00:12:02:03:03:7a:57:00:75:00:1a:00:12:\n      0e:47:4c:7b:7a:00:00:00:00:00:00:00:00:00:00:00:\n      0e:7b:0d:01:0d:00:3c:0f:17:9b:5d:0a:0a:19a:07:\n      0e:07:27:05:04:51:58:02:2a:05:07:03:0a:07:07:\n      71:0b:87:10:08:29:71:0d:13:05:0f:02:7b:2d:\n      0e:15:27:10:18:7a:03:14:44:03:3f:00:09:0a:02:\n      0a:3a:08:01:1a:28:73:8d:0d:00:72:1a:0a:0a:\n      0e:05:10:0a:28:10:0e:07:4d:44:00:c2:5a:52:\n      0a:33:00:00:7c:7b:42:0a:08:00:0e:02:48:07:74:\n      03:05:37:01:0a:08:0a:00:07:42:76:02:04:0a:\n      03:3a:07:03:2b:f1:0e:0a:00:04:07:03:7b:0f:08:\n      03:08:00:00:00:00:00:00:00:00:00:00:00:00:00:\n      7b:37:8b:11:1b:2e:5b:0b:0a:0e:00:00:0a:05:24:\n      05:08:00:00:00:00:00:00:00:00:00:00:00:00:00:\n      7a:7d:2a:72:7a:0a:41:c9:0f:3e:00:00:00:08:17:\n      05:2b:7b:10:00:02:7b:10:00:00:10:00:00:00:00:\n      02:43:45:0d:2f:0a:3e:34:2d:50:7c:63:23:c3:2a:\n      02:0e:81:01:00:00:00:00:00:00:00:00:00:00:00:\n      05:1a:01:00:0f:04:3b:80:53:0d:0e:7a:0a:7c:\n      07:7a:04:02:00:7b:19:7f:07:20:04:02:02:02:07:\n      00:15:7b:50:7c:21:1b:3a:10:01\n  Signature Algorithm: sha256WithRSAEncryption\n  X509v3 extensions (X509v3):\n    X509v3 Authority Key Identifier:\n      66:01:21:08:0a:25:27:00:07:01:02:49:04:00:00:00:64:0c:1e:0a:0f:08\n    X509v3 Subject Key Identifier:\n      61:5b:03:22:63:00:0c:0c:04:27:57:88:50:40:2b:43:08:49:15:1c\n    X509v3 Key Usage: critical\n      (digital) Signature, Key Encipherment\n    X509v3 Basic Constraints: critical\n      CA:FALSE\n    X509v3 Extended Key Usage:\n      TLS Web Server Authentication, TLS Web Client Authentication\n    X509v3 Certificate Policies:\n      Policy: 1.3.6.1.5.5.8.2.1.3.1.4\n      CRL: http://certs.gwdg.de/CP\n      Policy: 2.23.148.1.2.2\n    X509v3 CRL Distribution Points:\n\n      Full Name:\n      uri:http://certs.gwdg.de/certs/gwdg-generalservers/letsencryptserverCA.crt\n      Authority Information Access:\n      CA Issuers - URI:http://certs.gwdg.de/Certs/gwdg-organisations/letsencryptserverCA.crt\n      OCSP - URI:http://ocsp.sctigo.com\n    CT Precertificate SCTs:\n      Signed Certificate Timestamp:\n        Version  : v1 (0x01)\n        Log ID    : 9a:05:18:12:0a:96:f0:60:51:c2:01:02:0f:04:14:\n          8d:a4:c9:7b:0c:02:02:8b:9f:8e:07:4c:5a:5a:3a:74\n        Timestamp: Feb  2 87:27:56.203 2023 GMT\n        Extensions: none\n        Signature  : sha256WithRSAEncryption\n          3b:45:02:18:42:08:0c:18:0a:0e:18:0a:03:03:04:08:19:\n          0c:0a:19:08:0c:01:02:0c:13:64:02:09:20:0c:2b:0a:\n          00:03:0e:1a:82:21:00:23:9a:03:23:17:73:03:03:0c:\n          05:2b:18:10:03:03:00:20:2a:03:03:0a:01:00:0f:0a:0a:\n          08:54:73:4c:1c:1c:1c\n      Signed Certificate Timestamp:\n        Version  : v1 (0x01)\n        Log ID    : 0a:0e:0f:00:3f:05:00:22:04:08:0c:00:00:00:00:00:00:\n          01:71:04:08:11:0a:34:08:0a:01:30:00:00:07:08:0a\n        Timestamp: Feb  2 87:27:56.193 2023 GMT\n        Extensions: none\n        Signature  : sha256WithRSAEncryption\n          04:40:02:08:10:0a:00:00:03:7a:0a:17:0a:17:03:00:0f:0a:\n          20:09:03:07:07:0a:02:02:02:02:04:02:03:08:02:22:20:22:07:\n          06:04:02:00:02:05:13:71:0a:0a:0a:10:20:0a:7a:0a:01:\n          03:01:43:03:03:0c:00:40:50:40:08:f0:7c:15:0a:0c:15:\n          00:0a:00:00:00\n      Signed Certificate Timestamp:\n        Version  : v1 (0x01)\n        Log ID    : 61:03:08:6a:03:00:1a:c4:1c:1c:87:02:8a:03:13:02:\n          53:07:44:1c:0c:0c:0c:0c:0c:0c:0c:0c:0c:0c:0c:0c:0c:0c:0c:\n        Timestamp: Feb  2 87:27:58.148 2023 GMT\n        Extensions: none\n        Signature  : sha256WithRSAEncryption\n          2b:45:02:18:0a:0a:03:7a:0a:03:03:03:0f:0f:0c:07:07:\n          08:0f:0f:08:0a:00:00:18:07:01:81:00:08:08:01
```

The hps.vi4io.org Certificate

shell

```
echo | openssl s_client -connect hps.vi4io.org:443 2>/dev/null | \
openssl x509 --text
```


Certificate Standard: X.509

- Most common format
 - ▶ Separate from SSH and PGP
- From the “PKIX standards”
- Works out-of-the-box for TLS and HTTPS clients and servers
 - ▶ Good ecosystem support
- Arose from X.500 project in 1988
 - ▶ ITU-T (International Telecommunication Union Standards Body) tried to build a phone book

<https://www.rfc-editor.org/rfc/rfc5280>
<https://cabforum.org/baseline-requirements-documents/>

Certificate Encoding

- X.509 builds on ASN.1, another ITU-T standard
- ASN.1 (Abstract Syntax Notation One) is an encoding standard
 - ▶ Similar to JSON and Protobuf
- ASN.1 does not define binary representation
 - ▶ Its encoding rules can lead to incompatible binaries
- DER (Distinguished Encoding Rules) is an ASN.1 encoding format
 - ▶ Most commonly used for X.509
 - ▶ Sometimes also BER (Basic Encoding Rules) another ASN.1 encoding format
 - ▶ DER encoding/decoding handled by libraries

<https://www.rfc-editor.org/rfc/rfc5280>
<https://cabforum.org/baseline-requirements-documents/>

Problematic Standards

- Certificates may be DER format or something “fancier”
 - ▶ Raw DER is binary data
- PEM (Privacy Enhanced EMail) encapsulates certificates
 - ▶ Adds header and footer
 - ▶ May hold any DER certificate, private key, public key
 - ▶ Similar to PGP and S/MIME
 - ▶ Don't ask why it's called “EMail” that's another 4 RFC document
- PEM headers hold labels to describe payload
 - ▶ Attempt at standardization via RFC 7468
 - ▶ Labels can be inconsistent
- PEM file “typically” stored as .pem, .crt or .cer
 - ▶ Raw DER is commonly stored as .der

<https://www.rfc-editor.org/rfc/rfc7468>

Another Layer

- PEM might be included in an “envelope” with additional data
 - ▶ A “Certificate” might be a PEM file or an envelope with a PEM
- Envelopes are part of PKCS (Public Key Cryptography Standards)
 - ▶ Published by RSA (the company, not the algorithm)
- Most relevant are PKCS#7 and PKCS#12
 - ▶ Both can contain certificate chains
 - ▶ May hold PEM and raw DER, BER
 - ▶ May be stored as raw DER or as PEM
- PKCS#7 was rebranded as CMS (Cryptographic Message Syntax) by the IETF
 - ▶ Commonly stored as .p7b or .p7c and used in Java
- PKCS#12 commonly used with Microsoft products
 - ▶ Commonly stored as .pfx or .p12

<https://www.rfc-editor.org/rfc/rfc3447>

<https://www.rfc-editor.org/rfc/rfc2315>

<https://www.rfc-editor.org/rfc/rfc7292>

Key Encoding

- General pattern:
 - ▶ ASN.1 data structure
 - ▶ DER binary encoding
 - ▶ PEM headers
- Private keys commonly use PKCS#8
 - ▶ Can be encrypted using a passphrase
 - ▶ Uses another encoding standard within a PEM

A PEM in the Wild

- `hpcsa-course-vm-key.pem` was created for your cloud setup by OpenStack
- Try reading it with

```
shell
```

```
openssl x509 --text -noout -in hpcsa-course-vm-key.pem
```

- This fails, its a PEM but not a certificate

```
shell
```

```
openssl rsa -check -in hpcsa-course-vm-key.pem
```

- This prints your RSA private key

Summary

- X.509 is the standard for certificates
- DER is a common encoding used with it
- PEM encapsulates binary DER certificates for better usability on the web
- PKCS#7 and PKCS#12 encapsulate certs with additional certs and data
- Private keys are commonly PEM encoded

Overview

- By definition: Needs no certificates!
- Only goal: Bind names to public keys
 - ▶ `~/.ssh/authorized_keys` is a PKI
- Web PKI builds on top of certificates
 - ▶ Enables secure communication via the internet
- Anyone can create a public key and publish it
 - ▶ PGP (Pretty Good Privacy) concept
- On the web, what public keys and certificates to trust?

Trust Stores

- Trust Store contains preconfigured trusted root certificates
 - ▶ Usually preinstalled in your OS
 - ▶ Root certificates are self-signed
 - ▶ Who to trust to pick these?
- Microsoft, Apple, Mozilla (and partially Google) have trust store programs
- Over 100 CAs in most trust stores
 - ▶ Not all of them are morally "trustworthy"
 - ▶ Some governments operate CAs to spy on their citizens or impersonate websites
 - ▶ Some CAs were compromised in the past
- Trust stores are only as secure as the least secure CA

Attack Vector: Headset

- 2018 Sennheiser headset software installed a root certificate
 - ▶ Into the trust store of users
 - ▶ With the belonging private key hidden in the application
- Attackers would extract the private key
 - ▶ Use it to sign new web certificates
 - ▶ Systems with the Sennheiser software installed would accept them
- This is one example, there are more cases

<https://medium.com/asecuritysite-when-bob-met-alice/your-headphones-might-break-the-security-of-your-computer-4f304ed86611>

Improvements to Security

- Baseline for issuing CAs was made more strict
- RFC 6962 introduced CT (Certificate Transparency)
 - ▶ Impartial observers check for fraudulent certificates
 - ▶ Checks are recorded in the certificates
- Root CA certificates are not used for automatically signing certificates
 - ▶ Only kept on special air-gapped machines
 - ▶ Intermediate certs used for automatic signing (certificate chain)
- Do not disable certificate path validation
 - ▶ e.g., `curl -k`
 - ▶ Unless you know you can trust the server

<https://www.rfc-editor.org/rfc/rfc6962>

Inspecting Your Trust Store

- On Linux installed via ca-certificates package
- In /etc/ssl/certs
- Extract and inspect certs

```
shell
```

```
trust extract --format=x509-directory x509-certs && cd $_  
openssl x509 -noout -text -inform=der -in NAME.pem
```

- Look for
 - ▶ Issuer
 - ▶ Subject
 - ▶ Validity
 - ▶ x509 extensions

Table of contents

6 Practical Problems

7 Let's Encrypt

Getting a Certificate from a CA - In Theory

- 1 You create a key pair
 - 2 You submit your key with your data to a CA
 - 3 CA validates your data
 - 4 CA signs and returns a certificate
- After some time your certificate expires
- ▶ Replace it using above process

Getting a Certificate from a CA - In Practice

Solve two **hard** problems in computer science

- Cache invalidation
- Naming things

DNs (Distinguished Names)

- Certificate issuer/subject metadata field
 - ▶ Subject: YOUR_NAME
 - ▶ YOUR_NAME could be `my.domain.de`
- Historical artifact
- Instead only use CN (Common Name)
- List domains under SANs (Subject Alternative Names)
 - ▶ `DNS:my.domain.de`
 - ▶ Wildcards are possible `DNS:*.pages.gwdguser.de`
 - ▶ Be careful with wildcards

Generating a Key Pair

- Optimally, generate the key pair yourself
- Ensure security of the private key
- From key pair generate a CSR (Certificate Signing Request)
 - ▶ CSR is signed, can be freely shared
- Send CSR to a CA

Certificate Signing Request Validation

CSR Validation has 3 Options:

■ DV (Domain Validation)

- ▶ Check DNS name in WHOIS record
- ▶ Either send an email
- ▶ Or use ACME (Automatic Certificate Management Environment)
 - For a HTTP challenge
 - For a DNS TXT record check

■ OV (Organization Validation) and EV (Extended Validation)

- ▶ Builds on top of DV
- ▶ Associate legal entity with certificate
- ▶ Takes days or weeks to process

■ DV is good enough usually

<https://ietf-wg-acme.github.io/acme/draft-ietf-acme-acme.html>

Domain Validation Security

- What does successful DV mean?
 - ▶ Some entity was at one point in time able to either
 - Read email
 - Configure DNS
 - Serve an HTTP challenge
 - ▶ Relies on the security of these layers
 - ▶ Fraudulent certificates were issued
 - e.g., on an Amazon domain in 2018

<https://doublepulsar.com/hijack-of-amazons-internet-domain-service-used-to-reroute-web-traffic-for-two-hours-unnoticed-3a6f0dda6a6f>

Certificate Expiration

- Validity range on certs is very common
- Not a hard requirement
- Incorrect system clocks are a problem
 - ▶ Out of sync system clock might reject a valid cert
 - ▶ Systems without internal time reset to UNIX epoch 0 or 01.01.1970
- Private keys should have proper life cycle
 - ▶ Separate keys for certificate signing and encryption

Certificate Renewal

- No standard process
- Just replace the old cert with a new one
- Short lived certs are better for security
- Let's Encrypt defaults to 90 days

Certificate Revocation

- Certs might be unneeded
- Private keys can be compromised
- ⇒ Revoke the certificate
- Revoking X.509 certs is a mess
- Revocation status cannot be encoded in a certificate
- 2 systems in use
 - ▶ CRLs (Certificate Revocation Lists)
 - ▶ OCSP (Online Certificate Singing Protocol)

<https://scotthelme.co.uk/revocation-is-broken/>

CRLs (Certificate Revocation Lists)

- Defined in RFC 5280
- Long list of serial numbers of revoked certs
- Can be linked in a certificate
 - ▶ "Look for my serial number here, if I am on this list, I am revoked"
- What if the CRL endpoint is unreachable?
 - ▶ The certificate is accepted
- Lists are often cached
 - ▶ Propagating new revocations takes time

<https://www.ietf.org/rfc/rfc5280>

OCSP (Online Certificate Signing Protocol)

- Defined in RFC 2560
- Similar to CRL, endpoints can be queried for status of a cert
- Queries are for a specific certificate
- Privacy issue: The OCSP responder knows the websites you are visiting
- Solved by OCSP stapling
 - ▶ Owner of cert gets short-lived token from OCSP to validate cert
 - ▶ Might as well use short-lived certificates

<https://www.ietf.org/rfc/rfc2560>

Using a Certificate

- 1 Set up a HTTPs server
- 2 Point it to certificate file and public key
- 3 Done

Overview

- Non-profit company
- <https://letsencrypt.org/>
- Founded in 2012
- Provides a free service for acquiring certificates
 - ▶ Only supports automatic domain-validation
 - ▶ Over 300 mil (unexpired) certificates - November 2022

<https://letsencrypt.org/stats/>

Certbot

- Open-source tool
- Allows obtaining certificates from Let's Encrypt
- Can also install them via plugins
 - ▶ Support Apache, NGINX, etc.
- Automatically renew certificates
- Uses ACME for DV

<https://github.com/certbot/certbot>

Exercise

- Please work on the exercise now.